Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____ 2024 г.

ВЕБ-ПРИЛОЖЕНИЕ ДЛЯ АВТОМАТИЗАЦИИ РАСЧЕТА СТОИМОСТИ ПЛАСТИКОВЫХ ОКОН

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-09.03.01.2024.308/287 ПЗ ВКР

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____ В.А. Парасич
«___»_____2024 г.

Автор работы,
студент группы КЭ-405
_____А. А. Шабанов
«___»_____2024 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___»_____2024 г.

Челябинск-2024

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____2024 г.

ЗАДАНИЕ
на выпускную квалификационную работу бакалавра
студенту группы КЭ-405
Шабанову Артёму Александровичу,
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. Тема работы**: «**Веб-приложение для автоматизации расчета стоимости пластиковых окон» утверждена приказом по университету от «__» _____ 2024 г. №
2. Срок сдачи студентом законченной работы**:** 01 июня 2024 г.
3. Исходный данные к работе

Проект должен быть работать на разных версиях браузера и на всех типах устройств, а также разработан со следующим стеком технологий.

Серверная часть приложения должна быть написана на языке JavaScript с использованием программной платформы NodeJs. TypeScript заменит динамическую типизацию в JavaScript и предотвратит ошибки на этапе разработки проекта. В качестве серверного фреймворка используется NestJs в паре с документно-ориентированной не реляционной СУБД MondoDb.

Клиентская часть приложения должна быть написана с помощью HTML, CSS/SCSS, языка программирования JavaScript и TypeScript, библиотеки React, глобального хранилища Redux. Для тестирования приложения необходимо

использовать Jest и Storybook. Для контроля версий – Git. В качестве архитектуры был выбран подход FSD.

4. Перечень подлежащих разработке вопросов:
4.1. Аналитический обзор научно-технической, нормативной и методической литературы по тематике работы;
4.2. Разработка архитектуры и проектирование веб-приложения;
4.3. Разработка программного кода серверной и клиентской части веб-приложения;
4.4. Тестирование и отладка программного кода.

5. Дата выдачи задания**:** ___ декабря 2023 г.

Научный руководитель,
канд. юрид. наук, доцент,
В.А. Парасич _____ 2024 г.

Автор работы
Студент группы КЭ-405
А.А. Шабанов _____ 2024 г.

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	28.02.2023	
Разработка архитектуры и проектирование веб-приложения	03.03.2024	
Разработка программного кода серверной и клиентской части веб-приложения	24.03.2024	
Тестирование и отладка программного кода	07.04.2024	
Компоновка текста работы и сдача на нормоконтроль	08.04.2024	
Подготовка презентации и доклада	09.04.2024	

Руководитель работы _____/ *В.А. Парасич* /
Студент _____/ *А.А. Шабанов* /

АННОТАЦИЯ

Шабанов А.А. Выпускная Квалификационная работа «Веб-приложение для автоматизации расчета стоимости пластиковых окон» – ФГАОУ ВО «ЮУрГУ (НИУ)», КЭ-405, 50 с, библиогр. список – 19 наим., 2 прил.

В рамках выпускной квалификационной работы разрабатывается веб-приложение для расчета стоимости пластиковых окон. В ходе работы был проведен анализ предметной области и аналитический обзор литературы, а также были выбраны необходимые для разработки средства.

Далее описывается разработка архитектуры веб-приложения, выбор технологий и инструментов для его реализации, а также проектирование основных компонентов и модулей системы, определение требований к пользовательскому интерфейсу и бизнес-логике.

Следующая часть работы посвящена непосредственной разработке программного кода веб-приложения, описанию основных этапов разработки серверной и клиентской частей, реализации алгоритмов расчета стоимости окон и интеграции с сервисами и базами данных.

В заключительной части рассматриваются процессы тестирования и отладки разработанного программного кода, описываются методы тестирования, выявление и устранение ошибок, оценка производительности и надежности веб-приложения.

Результатом выпускной квалификационной работы является веб-приложение для расчета стоимости пластиковых окон, обладающее высокой функциональностью, удобным пользовательским интерфейсом и надежной реализацией.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Веб-разработка является неотъемлемой частью современного бизнеса, поскольку она позволяет компаниям создавать и поддерживать свои веб-ресурсы, такие как сайты, порталы и веб-приложения. Этот процесс включает в себя различные этапы, начиная с разработки логики и функциональной части, создания пользовательского интерфейса и заканчивая настройкой хостинга и покупкой доменного имени.

Веб-разработка делится на две основные отрасли: создание сайтов и веб-приложений. Сайты предоставляют информацию о бренде, услугах или товарах компании, в то время как веб-приложения служат для взаимодействия с пользователями или сотрудниками.

Существует множество причин, по которым бизнес может обратиться к услугам веб-разработки. Среди них повышение престижа и узнаваемости бренда, расширение каналов продаж, создание платформы для общения с клиентами или сбора данных, а также разработка дополнительных веб-приложений для улучшения пользовательского опыта.

Процесс веб-разработки может быть сложным и длительным, но он позволяет компаниям оптимизировать свои бизнес-процессы, улучшить взаимодействие с клиентами и увеличить прибыль.

Одним из решений может стать разработка веб-калькулятора для подсчета стоимости окон. Актуальность данной темы обусловлена несколькими факторами:

1. Экономическая выгода: Веб-калькулятор позволит клиентам быстро и удобно рассчитать стоимость окон учитывая их индивидуальные предпочтения и

требования. Это поможет сэкономить время и средства, связанные с вызовом замерщиков и обсуждением цены с менеджерами.

2. Улучшение имиджа компании: Наличие современного и функционального веб-калькулятора, который позволяет быстро и точно рассчитать стоимость окон, может положительно сказаться на имидже компании и вызвать доверие у потенциальных клиентов.

3. Повышение конкурентоспособности: Предприятия, предоставляющие услуги по установке окон, смогут предложить своим клиентам более удобные и выгодные условия, что в свою очередь может привлечь новых заказчиков и увеличить прибыль.

4. Оптимизация работы компании: Разработка веб-калькулятора позволит сэкономить время сотрудников, занимающихся обработкой заказов, а также уменьшить количество ошибок и недопонимания между клиентами и менеджерами.

Таким образом, разработка веб-калькулятора для расчета стоимости окон является актуальной задачей, решение которой позволит улучшить работу предприятий данной сферы, повысить их конкурентоспособность и привлечь больше клиентов.

Целью ВКР является разработка веб-приложения для автоматизации расчета стоимости пластиковых окон для компании ООО ТПК «ИНВЕСТ-ТРЕЙД». Основная задача веб-приложения заключается в обеспечении эффективной и емкой информации клиента о компании, ее деятельности, товарах и услугах. Оформление заказа с помощью онлайн калькулятора и корзины. Расчет стоимости пластиковых окон с учетом параметров пользователя. Обеспечение руководства над веб-приложением при помощи панели управления администратора, где возможно редактирование страницы и данных товара. Реализация личного кабинета, авторизации для клиента и администратора.

Для выполнения этой цели необходимо последовательно решить следующие задачи:

1) аналитический обзор научно-технической, нормативной и методической литературы по тематике работы;
2) разработка архитектуры и проектирование веб-приложения;
3) разработка программного кода серверной и клиентской части веб-приложения;
4) тестирование и отладка программного кода;

ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Обзор научной литературы

На данный момент существует множество различных оконных компаний, который внедрили в свой сайт онлайн калькулятор. Рассмотрим некоторые из них, связанные с темой работы.

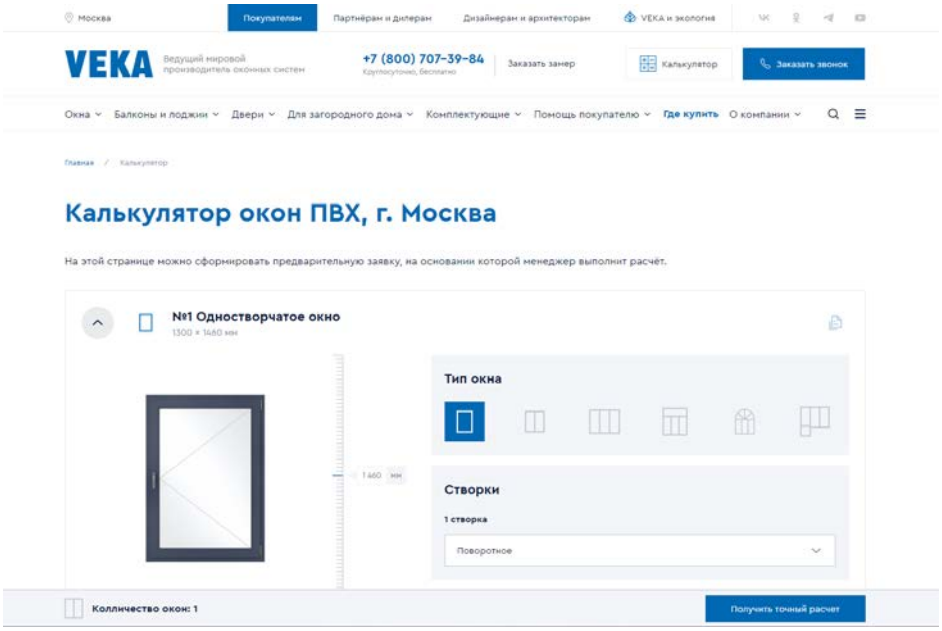

Рисунок 1 – «https://www.veka.ru/calc» скриншот страницы калькулятора.

«https://www.veka.ru/» [1] – компания «VEKA». На рисунке № 1 изображен онлайн калькулятор для подсчета стоимости окон ПВХ. У данного продукта есть несколько достоинств. Во-первых, мы можем выбрать сразу несколько окон для подсчета и скопировать уже настроенную конфигурацию окна. Это позволяет сэкономить время заказчика и интуитивно быстро оформить заявку на расчет. Во-вторых, при выборе типа окна, автоматически подгружаются настройки и функции, характерные только для данного типа. Без сомнений, это позволяет избегать ошибок в расчетах и помогает сотрудникам предприятия избежать ручного перерасчета при ошибке выбора пользователя.

К недостаткам данного тренажера можно отнести: получение точного расчета только после оформления заявки и никакого результата после настройки калькулятора. Пользователь, который решил воспользоваться онлайн расчетом зачастую не имеет опыта в данной сфере и хочет быстрого решения здесь и сейчас. Долгое ожидание результата быстро отобьет у клиента желание задержаться на данном сайте. Вместо этого можно предложить готовый вариант окна, который есть в наличии в ассортименте и будет совпадать по характеристикам, выбранным пользователем. И после этого уже отправить клиенту точный расчет стоимости.

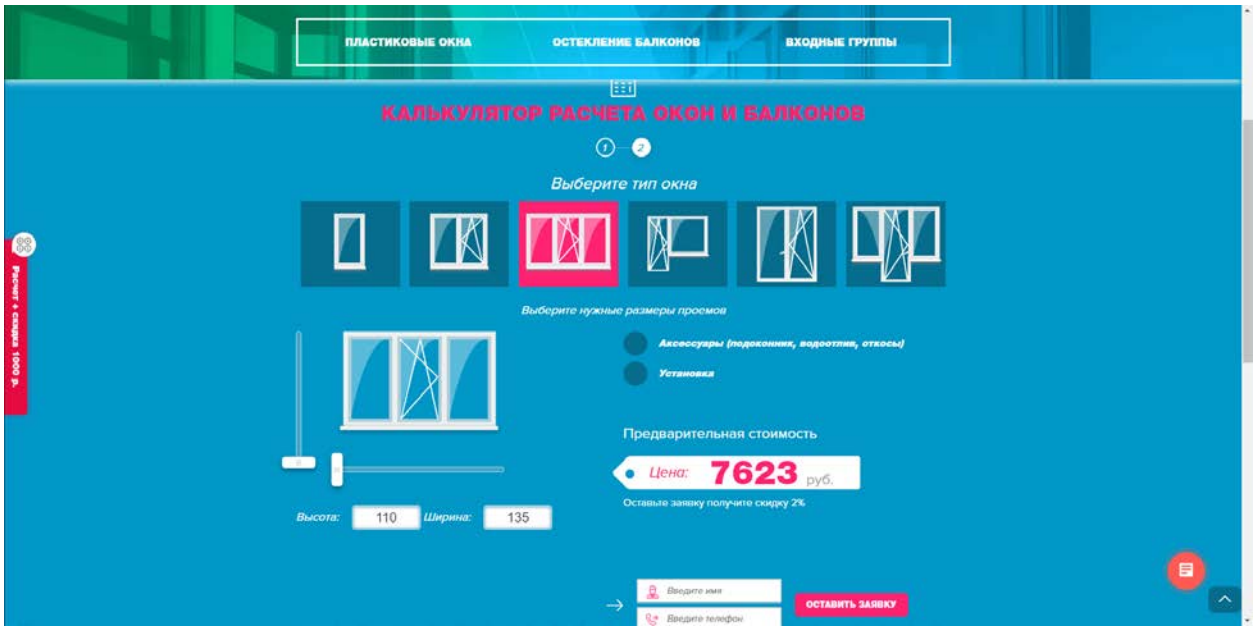

Рисунок 2 – «https://oknadarom74.ru/calc/» скриншот страницы калькулятора.

«https://oknadarom74.ru» [2] – компания «ОКНА ДАРОМ». На рисунке № 2 изображен онлайн калькулятор для подсчета стоимости окон ПВХ. Из преимуществ можно выделить лишь моментальный автоматический подсчет точной стоимости.

Однако недостатков данный калькулятор имеет куда больше. Небольшой выбор функционала, отсутствие возможности посчитать больше одного окна, скопировать настройки конфигурации, получить точную модель профиля и фурнитуры для данных настроек. В данном случае, такое решение не даст ответа

для клиента, имеющего точные запросы, например, на тип дома, установки отлива, возможности выбрать бюджетный или дорогой вариант окна. И от подсчета стоимости окна не остается толку, ведь клиенту придется самому считать все остальные аспекты установки и дополнений.

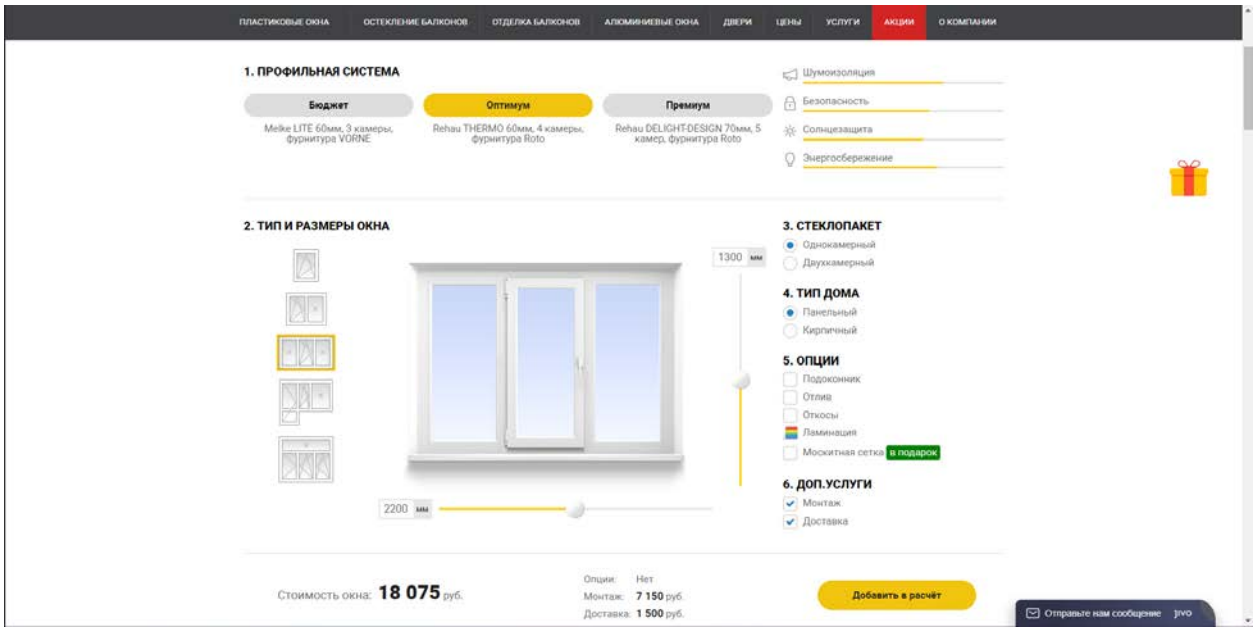

Рисунок 3 – «https://www.okna-servise.com/calculator/»
скриншот страницы калькулятора.

«https://www.okna-servise.com» [3] – компания «OKNA SERVICE». На рисунке 4 изображен онлайн калькулятор для подсчета стоимости окон ПВХ.

Разберем преимущества данного сервиса. Во-первых, клиент имеет возможность выбрать тип профильной системы. От бюджетной настройки до премиума качества. Справа от выбора системы можно увидеть диаграмму разницы в характеристиках шумоизоляции, безопасности, солнцезащиты и энергосбережения. Переключая между качеством, покупатель может сравнить их и понять, чего он может лишиться при покупке дешевой системы или стоит ли ему переплачивать за более дорогой вариант. Во-вторых, это мгновенный подсчет стоимости, которая делится на стоимость окна, отдельных опций, монтажа и учета доставки. Заказчик будет точно знать из чего состоит его сумма, будь это доставка или размер окна.

Из недостатков можно отметить лишь неудобное использование данного калькулятора. Все секции и этапы выбора расположены друг под другом, которые скрываются при прокрутке страницы вверх или вниз. Правильным вариантом будет расположить размеры, стоимость и опции на одном экране сразу и динамически отображать их.

Таблица 1 – Сравнение аналогов

	Компания «VEKA»	Компания «ОКНА ДАРОМ»	Компания «OKNA SERVICE»
Копирование конфигурации	+	-	-
Выбор типа профильной системы	-	-	+
Диаграмма характеристик	-	-	+
Мгновенный подсчет	-	+	+
Калькулятор на одном экране	-	+	-
Выбор подсчет нескольких профилей	+	-	+
Загрузка характеристик профиля	+	-	+
Выбор подходящего	-	-	-

профиля по характеристикам			

1.2 Инструменты разработки

Для разработки дизайна сайта был выбран сервис Figma.

Figma – графический редактор для создания прототипов сайтов и приложений.

Для создания веб-приложения выбраны следующий программные средства: HTML, CSS, JavaScript, TypeScript, React.js, Webpack, Redux Toolkit, Node.js, Nest.js, MongoDB. Для тестирования были выбраны такие библиотеки, как Jest и Storybook. Для проектирования приложения будем использовать архитектуру FSD. Чтобы отслеживать историю изменений и контролировать версии приложения, используем Git. В качестве среды для разработки использован Visual Studio Code. Для реализации системы управления веб-приложением была создана панель управления администратора.

MongoDB – это ориентированная на документы база данных NoSQL с открытым исходным кодом, которая использует для хранения структуру JSON. Модель данных MongoDB позволяет представлять иерархические отношения, проще хранить массивы и другие более сложные структуры [4].

Nest JS — это платформа для создания эффективных масштабируемых серверных программ на языке Node.js. Она использует JavaScript и полностью поддерживает TypeScript. Nest JS предлагает готовую архитектуру приложений, которая позволяет разработчикам создавать легко тестируемые, масштабируемые, слабо связанные и простые в обслуживании приложения. [5].

React — это JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов. Она была создана компанией Facebook

и используется для создания одностраничных и мобильных приложений. React обеспечивает высокую скорость разработки, простоту и масштабируемость. [6].

TypeScript – это расширенная версия языка JavaScript, изначально созданная в Microsoft для разработки крупных приложений. TypeScript помогает избавиться от типичных проблем JavaScript: ошибок типов в рантайме и неконтролируемо разрастающегося кода, сигнатуры функций которого находятся в лучшем случае в памяти разработчика, а в худшем и вовсе утрачены. [7].

Webpack — это сборщик модулей JavaScript с открытым исходным кодом. Он принимает модули с зависимостями и генерирует статические ресурсы, представляющие эти модули. Webpack также позволяет использовать модульный подход для разработки веб-приложений. [8].

Redux – это инструмент для управления состоянием данных и пользовательским интерфейсом в приложениях JavaScript с большим количеством сущностей [9].

JavaScript – мультипарадигменный язык программирования. Наиболее широкое применение находит как язык сценариев веб-страниц. Он позволяет перехватывать события и выполнять различные действия. [10].

HTML («язык гипертекстовой разметки»). Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства. [11].

CSS («каскадные таблицы стилей») – формальный язык описания внешнего вида страниц, написанного с использованием языка разметки. [12].

Node.js – это программная платформа, основанная на движке V8, которая превращает JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через API, подключать внешние библиотеки и обеспечивать вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но также может использоваться для разработки десктопных приложений и программирования микроконтроллеров. [13].

Git – это система контроля версий, которая помогает отслеживать историю изменений в файлах. Она используется программистами для совместной работы над проектами. Git позволяет возвращать к предыдущей версии кода, просматривать историю изменений, работать параллельно над проектом и делать резервные копии кода. [14].

RTK Query — это инструмент для создания сервисов запросов на сервер в React с использованием Redux Toolkit. Он позволяет создавать удобные интерфейсы для отслеживания состояний запросов и уменьшает количество однообразного кода. [15].

FSD (Feature Sliced Design) — это архитектурная методология для проектирования frontend-приложений. Она основана на принципах SOLID, GRASP, DDD, Separation of Concerns, Vertical Slices и Public API. FSD разделяет проект по предметным областям согласно бизнес-логике и обеспечивает единообразие, контролируемое переиспользование логики, устойчивость к изменениям и рефакторингу, а также упрощает масштабирование проекта и команды. [16].

Jest — это фреймворк для тестирования JavaScript, разработанный и поддерживаемый компанией Facebook. Он используется для модульного тестирования, то есть проверки отдельных функций или блоков кода. Jest обладает такими особенностями, как нулевая конфигурация, моментальные снимки и изолированные тесты. [17].

Storybook — это инструмент JavaScript для организации пользовательских интерфейсов, который делает процессы разработки компонентов, тестирования и создания документации более эффективными и простыми. Он поддерживает множество фреймворков и библиотек для веб-приложений, включая React, Vue и Angular. Storybook пропагандирует подход Component-Driven Development (CDD), согласно которому каждая часть пользовательского интерфейса — это компонент. [18].

Figma — это онлайн-редактор, который используют для проектирования интерфейсов, создания макетов сайтов, мобильных приложений, презентаций, иллюстраций, логотипов и анимации. Figma удобна тем, что макет можно

просматривать и редактировать онлайн, а изменения видны сразу всем участникам проекта. [19].

В результате обзора были выделены основные задачи для реализации веб-приложения. На этапе проектирование мы должны создать качественный дизайн и мобильную адаптацию, потому что в современном мире многие используют различные устройства для просмотра информации. Так же очень важной частью является интуитивный интерфейс и хорошо подобранные цвета бренда компании. Веб-приложение должен быть быстрее в сравнении с конкурентами, для этого мы будем использовать современный фреймворк и максимально оптимизировать наш код. Так же для удобства разработки был выбран стек технологий MNRN (MongoDb, NestJs, ReactJs, NodeJs), который использует один язык программирования – JavaScript и типизация на TypeScript.

ГЛАВА 2. РАЗРАБОТКА АРХИТЕКТУРЫ И ПРОЕКТИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЯ

2.1. Функциональные и нефункциональные требования

Перед разработкой программного продукта необходимо сформулировать требования к нему. Эти требования можно разделить на функциональные и нефункциональные. К функциональным требованиям относятся действия, которые система должна выполнять. Это описание того, что нужно реализовать в процессе работы. Нефункциональные требования описывают как именно будет выполнена реализация и особенности эксплуатации.

Проектируемая система должна соответствовать следующим функциональным требованиям.

1. Пользователь должен иметь возможность авторизации на сайте.

2. Авторизация на сайте должен включать в себя валидацию полей, маску для номера телефона, обязательные и необязательные поля, защиту от спама ботов и соглашение с обработкой персональных данных.

3. Пользователь должен иметь возможность переходить по ссылкам меню.

4. Пользователь должен иметь возможность выбирать тип окна.

5. Поиск по сайту или товару должен быть динамический без перезагрузки страницы.

6. Пользователь должен иметь возможность только после авторизации на сайте получить результат расчета калькулятора.

7. Количество одновременных расчетов не должно превышать 4.

8. При отсутствии товара после поиска должна быть выведена картинка с ошибкой поиска.

9. Пользователь должен перейти на страницу с ошибкой после перехода по неправильному или несуществующему URL.

10. Пользователь должен иметь возможность добавить товар в корзину или в избранный товар.

11. Пользователь должен иметь возможность на странице корзины удалить один элемент, увеличить или уменьшить количество товара, полностью удалить всю корзину и оформить заказ.

12. Пользователь должен получать уведомление на сайте об успешном или неуспешном действии.

13. Пользователь должно только после авторизации иметь возможность оформить заказ через корзину на сайте.

14. Пользователь не должен иметь доступ к страницам администратора и его функционалу.

15. Веб-приложение должно быть адаптивным под разные размеры экрана.

Проектируемая система должна удовлетворять следующим нефункциональным требованиям.

1. Макет дизайна для веб-приложения должен быть разработан с использованием Figma.

2. Клиентская часть веб-приложения должна быть разработана с использованием HTML, SCSS, React.js, TypeScript, Webpack, JavaScript и Redux.

3. Серверная часть веб-приложения должна быть разработана с использованием REST API, Nest.js и MongoDB.

2.2. Диаграмма вариантов использования

Для проектирования системы был использован язык графического описания UML (англ. "Unified Modeling Language") – стандартизированный язык моделирования при проектировании программ. В соответствии с требованиями была построена диаграмма вариантов использования (англ. use-case diagram) – диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен пользователям (рисунок 4). В нашем случае диаграмма отражает модель взаимодействия актера «Пользователь» с системой.

Рисунок 4 – Диаграмма вариантов использования.

Основные актеры, которые взаимодействуют с системой.

На диаграмме представлен 1 актер – пользователь.

Пользователь – это клиент компании Инвест-трейд, который посетил страницу онлайн калькулятора, чтобы посчитать стоимость профиля, установки и доставки окна.

Краткое описание вариантов использования.

Добавить новое окно: чтобы пользователь мог рассчитать больше одного окна расчет, ему необходимо добавить новое окно.

Удалить окно: Пользователь может удалить дополнительные расчеты окон, но не может удалить все расчеты. Всегда остается один основной.

Выбрать настройки окна: чтобы пользователь получил расчет, ему необходимо выбрать настройки окна. Для этого клиент может выбрать такие опции как:

- Выбрать тип окна
- Ввести ширину с клавиатуры
- Ввести ширину с помощь скролла
- Ввести высоту с клавиатуры
- Ввести высоту с помощь скролла
- Выбрать расположение ручки
- Выбрать режим створки
- Выбрать бюджет
- Выбрать место установки
- Выбрать повышенную прочность стекла
- Выбрать взломостойкость
- Выбрать зеркальную тонировку
- Выбрать детские замки

Скопировать конфигурацию: Пользователь может скопировать расчет без повторного ввода данных.

Получить расчет и подобрать профиля: Пользователь после успешного расчета получает стоимость и тип предлагаемого профиля, а также стоимость доставки и услуг компании.

Отчистить расчет: Пользователь после настройки конфигурации может отчистить расчет выбранного окна, чтобы начать сначала.

Спецификация вариантов использования приведена в таблицах 2-7.

Таблица 2 – Спецификация добавления нового окна

UseCase: Добавить новое окно
ID: 1
Аннотация: Добавление нового окна в расчет
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловия:

1. Пользователь находится на странице Калькулятора
Основной поток: 1. Вариант использования начинается, когда Пользователь нажимает кнопку добавить новое окно. 2. Система проверяет количество созданных новых окон в расчете. 3. Если окон меньше 4. 3.1. Объект пустого нового окна добавляется в массив. 3.2. На экране появляется сообщение об успешном создании нового окна. 3.3. Система предоставляет доступ к действиям с данными. 4. Если окон больше 4. 4.1. На экране появляется сообщение о максимально возможном количестве окон в одном расчете.
Постусловия: 1. Открывается доступ к «Удалить окно» и «Выбрать настройки окна»
Альтернативные потоки: Нет

Таблица 3 – Спецификация удаления окна

UseCase: Удалить окно
ID: 2
Аннотация: Удаление окна из расчета
Главные актеры: Пользователь
Второстепенные актеры: нет
Предусловия: 1. Пользователь находится на странице Калькулятора 2. Пользователь создал новое окно
Основной поток: 1. Вариант использования начинается, когда Пользователь нажимает на кнопку удалить окно. 2. На экране появляется уведомление о подтверждении удаления окна из расчета. 3. Если пользователь подтвердил удаление. 3.1. Система удаляет выбранное окно по уникальному номеру из массива. 3.2. Система закрывает уведомление о подтверждении удаления окна из расчета.

3.3. На экране появляется сообщение об успешном удалении окна. 4. Если пользователь отменил удаление. 4.2. Система закрывает уведомление о подтверждении удаления окна из расчета.
Постусловия: 1. БД содержит новую запись. 2. В панели администратора появился новый пользователь.
Альтернативные потоки: Нет

Таблица 3 – Спецификация копирования конфигурации

UseCase: Скопировать конфигурацию
ID: 3
Аннотация: Скопировать конфигурацию для нового окна
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловия: 1. Пользователь находится на странице Калькулятора
Основной поток: 1. Вариант использования начинается, когда Пользователь нажимает на кнопку скопировать конфигурацию 2. Система проверяет количество созданных новых окон в расчете. 3. Если окон меньше 4. 3.1. Объект пустого нового окна добавляется в массив. 3.2 Объекту нового окна присваиваются настройки и характеристики скопированного окна. 3.2. На экране появляется сообщение об успешном копировании окна. 3.3. Система предоставляет доступ к действиям с данными. 4. Если окон больше 4. 4.1. На экране появляется сообщение о максимально возможном количестве окон в одном расчете.
Постусловия: 1. Открывается доступ к «Удалить окно» и «Выбрать настройки окна»
Альтернативные потоки: Нет

Таблица 4 – Спецификация отчистки расчета

UseCase: Отчистить расчет
ID: 4
Аннотация: Отчистить характеристики и настройки выбранного расчета
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловия: 1. Пользователь находится на странице Калькулятора 2. Пользователь создал новое окно 3. Пользователь выбрал хотя бы одну настройку для расчета
Основной поток: 1. Вариант использования начинается, когда Пользователь нажимает на кнопку отчистить расчет 2. Система присваивает выбранному окну стандартные настройки или пустое значение 3. На экране появляется сообщение об успешной отчистке расчета
Постусловия: нет
Альтернативные потоки: Нет

Таблица 5 – Спецификация выбора типа окна

UseCase: Выбрать тип окна
ID: 5
Аннотация: Выбрать тип окна
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловия: 1. Пользователь находится на странице Калькулятора
Основной поток: 1. Вариант использования начинается, когда Пользователь выбирает расчет. 2. Пользователь выбирает тип окна 3. Если тип окна 3.1. Система меняет иконку окна. 3.2 Система меняет большую схему окна. 3.2. На экране появляется кнопка далее.

4. Если НЕ тип окна 4.1. Система меняет иконку окна. 4.2 Система меняет большую схему окна. 4.2. На экране пропадает кнопка далее.
Постусловия: нет
Альтернативные потоки: Нет

Таблица 6 – Спецификация выбора параметров окна

UseCase: Выбрать параметры окна
ID: 6
Аннотация: Выбрать характеристики и настройки окна
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловия: 1. Пользователь находится на странице Калькулятора 2. Пользователь выбрал тип окна 3. Пользователь нажал на кнопку далее
Основной поток: 1. Вариант использования начинается, когда Пользователь нажимает на кнопку далее. 2. Пользователь выбирает параметры окна 3. Если ширина или высота окна меньше минимального значения или больше максимального значения 3.1. Система показывает ошибку о невозможных данных 3.2 Система присваивает минимальное значение. 3.2. На экране изменяются данные. 4. Если ширина или высота окна меньше максимального значения и больше максимального значения 4.1. Система меняет иконку окна. 4.2 Система меняет большую схему окна. 4.3 Система меняет данные расчета окна. 4.4. На экране изменяются данные. 4.5. На экране появляется кнопка далее. 5. Если расположение ручки или установка или бюджет 5.1. Система меняет иконку окна. 5.2 Система меняет большую схему окна. 5.3 Система меняет данные расчета окна. 5.4. На экране изменяются данные.

Постусловия: нет
Альтернативные потоки: Нет

2.3. Диаграмма активности

Одной из основных задач онлайн калькулятора является возможность добавление нового окна в расчет. Диаграмма активности описывает данный алгоритм (рисунок 5).

В начале пользователь нажимает на кнопку добавить новое окно. В случае, если количество расчетов больше 4 операция не начнет свое выполнение и выдаст ошибку. При отправке изменяется состояние калькулятора, проверяя наличие идентичных расчетов в калькуляторе. Если расчеты одинаковы, то он увеличит количество на один. Иначе добавит пустой расчет в систему и подсчитает сумму с базовой конфигурацией.

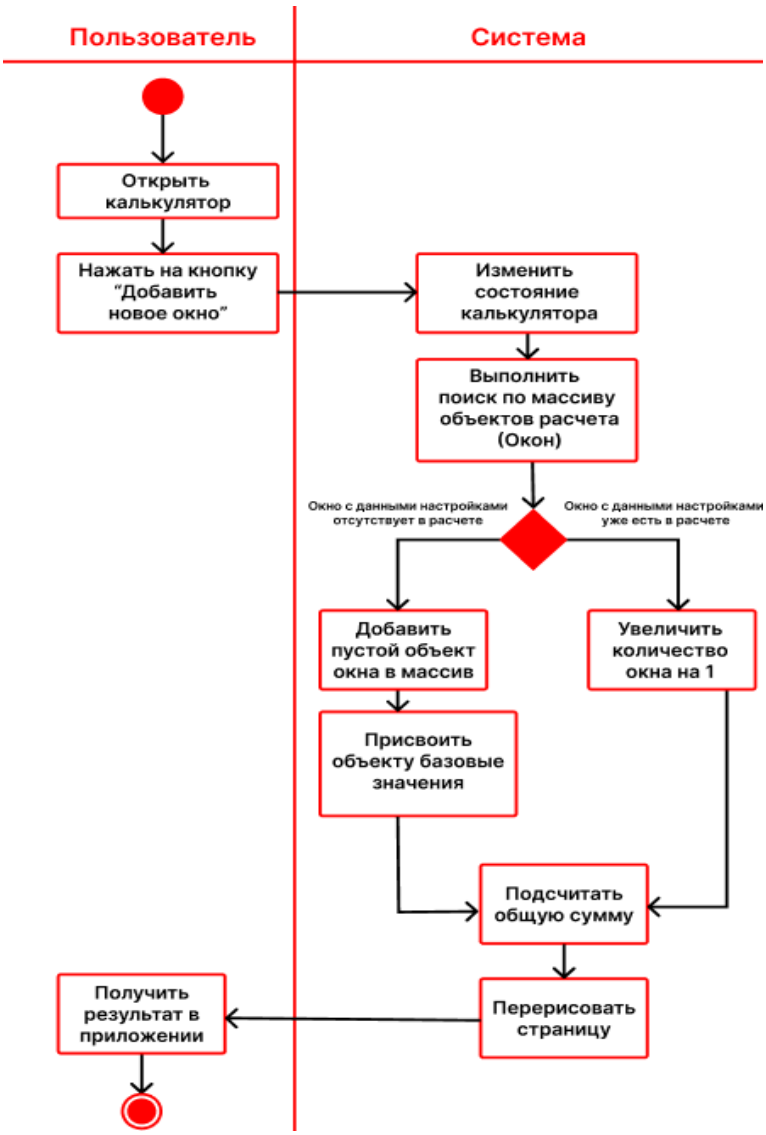

Рисунок 5 – Диаграмма активности добавления нового окна

2.4. Структура базы данных

База данных – это структурированный набор всех данных, расположенных на сайте. Для разрабатываемого веб-приложения используются MongoDB Compass. Это приложение с открытым кодом, предоставляющее собой интерфейс для администрирования СУБД MongoDB. Она позволяет удобно переключаться между таблицами, просматривать их и редактировать без использования команд.

В таблице 7 представлена структура таблиц и ее краткое описание.

Таблица 7 – описание таблиц товара БД.

Структура таблицы	**Данные таблицы**

export class ProductModel {
slug: string;
poster: string;
title: string;
is_available: boolean;
brand: string;
description: string;
rating: number;
view_count: number;
details: Settings[]
category: string;
price: number;
adds: Adds[];
isSendTelegram: boolean;
}
	Товар

export class Settings {
name: string
value: string
}
	Характеристики товара

export class Adds {
title: string;
description: string;
price: number;
poster: string;
}
	Опции товара

2.5. Проектирование дизайна веб-приложения

Для разработки UX/UI дизайна сайта было проведено совещание с заказчиком для уточнения желаний, разбора похожий по структуре макетов и определения целевой аудитории веб-сайта. После переговоров было принято решение полностью отойти от старого дизайна для разработки фирменного стиля компании. Цвет должен быть спокойным, вызывать ассоциацию с деятельностью компании. Шрифт и его размер выбран для выделения значимой и уменьшения

вторичной информации для пользователя, а также для удобства чтения на всех устройствах.

С помощью сервиса Figma, на основе технического задания компании, был разработан макет сайта. На рисунках 1-7 приложения А представлены макеты всех блоков сайта.

2.6. Архитектура FSD

FSD (Функциональная системная декомпозиция) - это архитектурный подход, который фокусируется на разделении приложения на функциональные модули, основанные на предметной области.

В FSD-архитектуре проект состоит из уровней (layers), каждый уровень состоит из срезов (slices), а каждый срез состоит из сегментов (segments) (Рисунок 6).

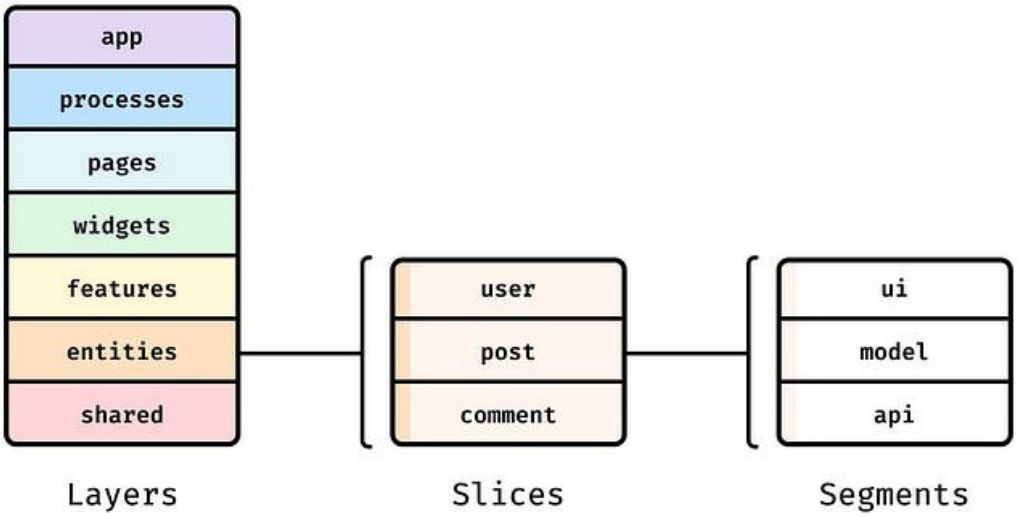

Рисунок 6 – схема уровней архитектуры.

Ниже представлена иерархия уровней при разработке онлайн калькулятора (Рисунок 7).

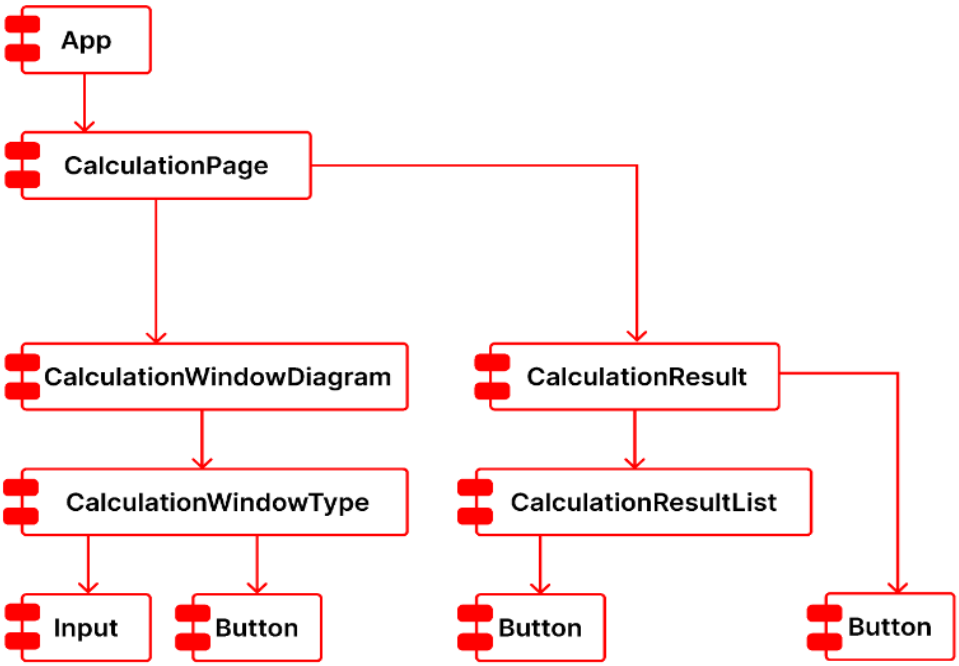

Рисунок 7 – схема уровней архитектуры при разработке калькулятора.

Подводя итог по второй главе, можно сказать, что были определены функциональные и нефункциональные требования к системе. На основе этих требований была построена диаграмма вариантов использования, определены основные актеры, взаимодействующие с системой, а также приведены краткое описание и спецификация вариантов использования.

ГЛАВА 3. РАЗРАБОТКА ПРОГРАММНОГО КОДА СЕРВЕРНОЙ И КЛИЕНТСКОЙ ЧАСТИ ВЕБ-ПРИЛОЖЕНИЯ

3.1. Верстка макета и реализация функционала на клиенте

Навигация на страницу онлайн калькулятора.

Удобная навигация на сайте - это ключевой элемент, который влияет на успех и востребованность любого веб-ресурса. От того, насколько интуитивно понятна и легкодоступна структура сайта, зависит, сможет ли пользователь быстро найти нужную ему информацию и завершить целевые действия.

Эффективная навигация помогает создать положительный пользовательский опыт и повысить конверсию сайта. Она позволяет посетителям с легкостью перемещаться по разделам, быстро ориентироваться в контенте и получать доступ к нужным данным или услугам.

В современном динамичном онлайн-пространстве, где внимание пользователей рассеяно, удобная навигация становится одним из ключевых факторов, влияющих на общее восприятие сайта, его запоминаемость и лояльность аудитории. Она напрямую связана с достижением бизнес-целей, будь то увеличение продаж, формирование базы подписчиков или повышение узнаваемости бренда.

Поэтому разработке эффективной системы навигации необходимо уделять самое пристальное внимание на этапе проектирования и регулярно тестировать ее работоспособность. Только в этом случае сайт станет максимально удобным и полезным для своих посетителей.

В нашем случае мы добавим широкой блок для перехода на страницу для расчета прямо на главной странице веб-приложения (Рисунок 8), а также разместил ссылку в меню быстрой навигации (Рисунок 9).

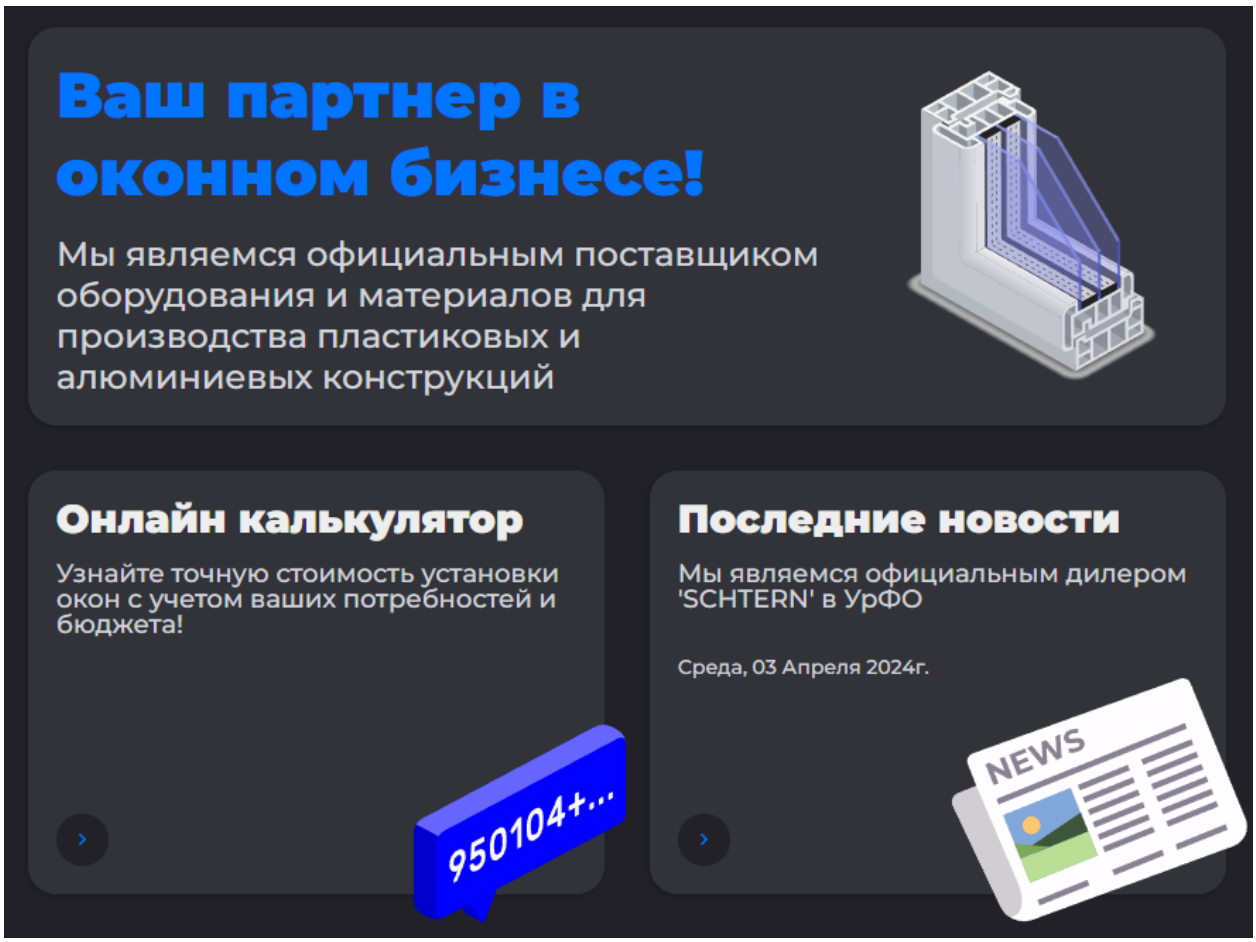

Рисунок 8 – Блок-ссылка калькулятора на главной странице

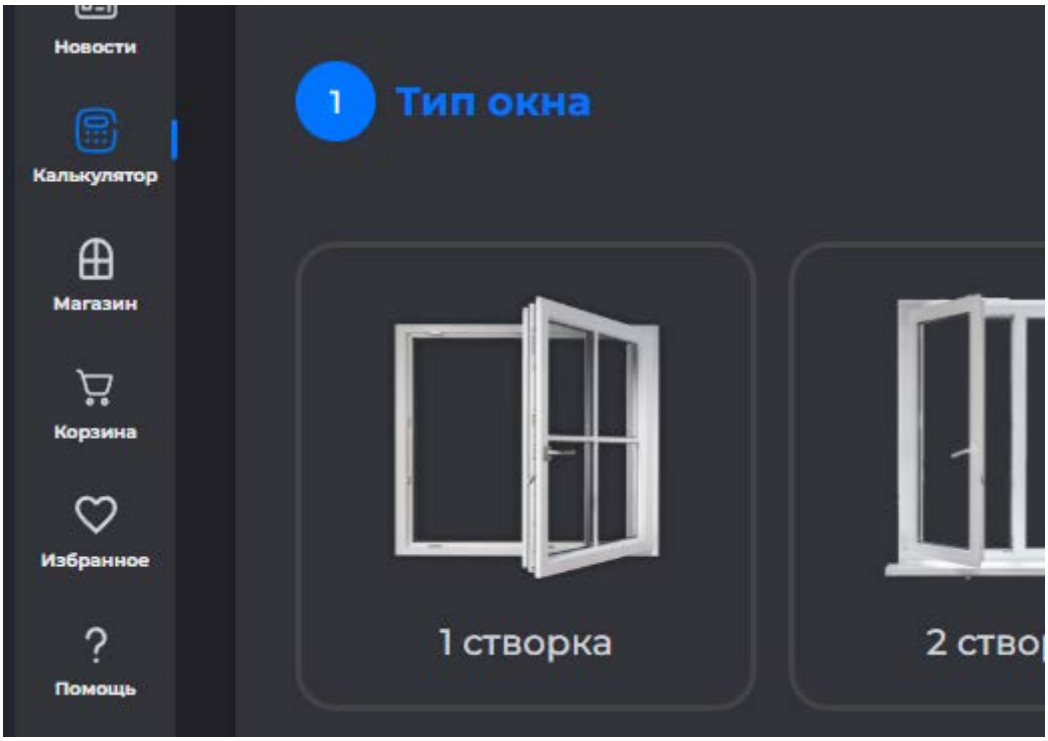

Рисунок 9 – Ссылка в меню быстрой навигации

Блоки этапов расчета

При разработке интерактивного интерфейса пользователя в React одной из важных задач является реализация возможности расчета стоимости продукта или услуги. В случае с оконными конструкциями, эта функциональность позволяет посетителям сайта или приложения быстро получать актуальную информацию о ценах на окна с учетом их индивидуальных параметров и потребностей.

Чтобы создать эффективный и удобный инструмент для расчета стоимости окон в React, необходимо четко структурировать процесс и разбить его на логические шаги. Каждый из этих этапов будет отвечать за сбор и обработку определенных данных, а в совокупности они сформируют целостный и интуитивно понятный пользовательский опыт.

В нашей реализации расчет разделен на три этапа. На рисунке 10 изображен этап выбора типа окна. В зависимости от типа, будут отображаться доступные настройки системы в следующем этапе параметры (рисунок 11). На 3 этапе выбора дополнительной комплектации расчет будет завершен (рисунок 12).

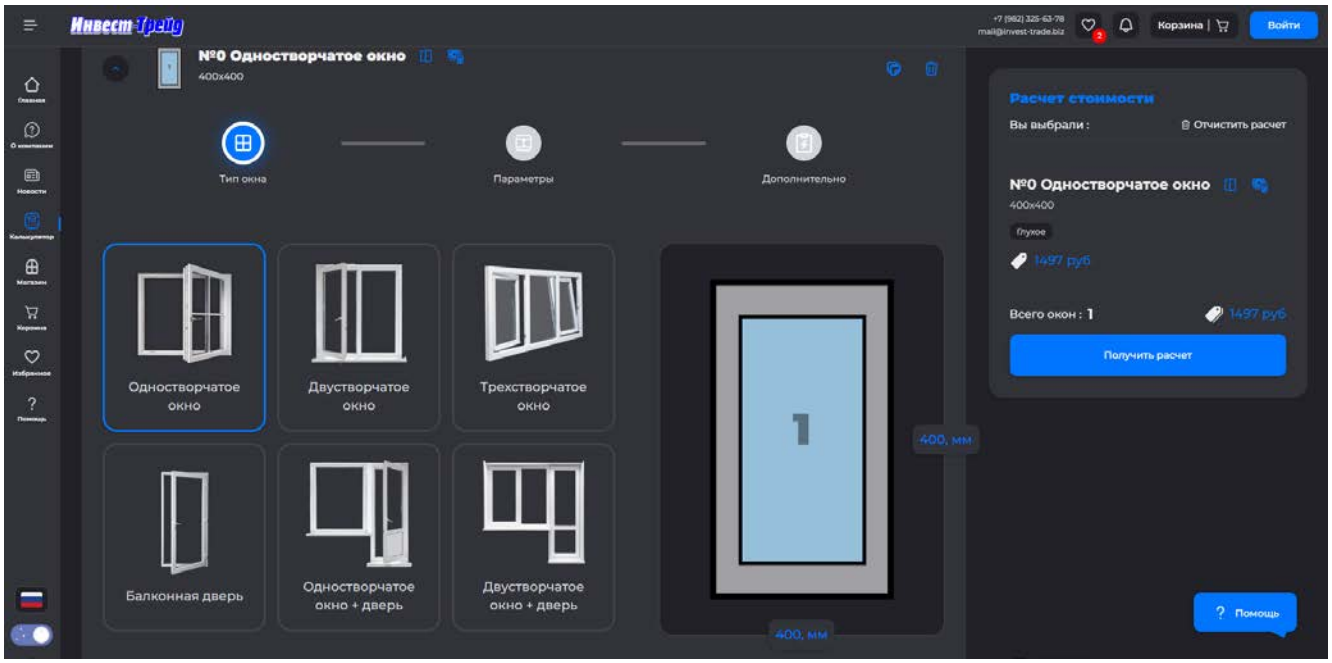

Рисунок 10 – Этап выбора типа окна

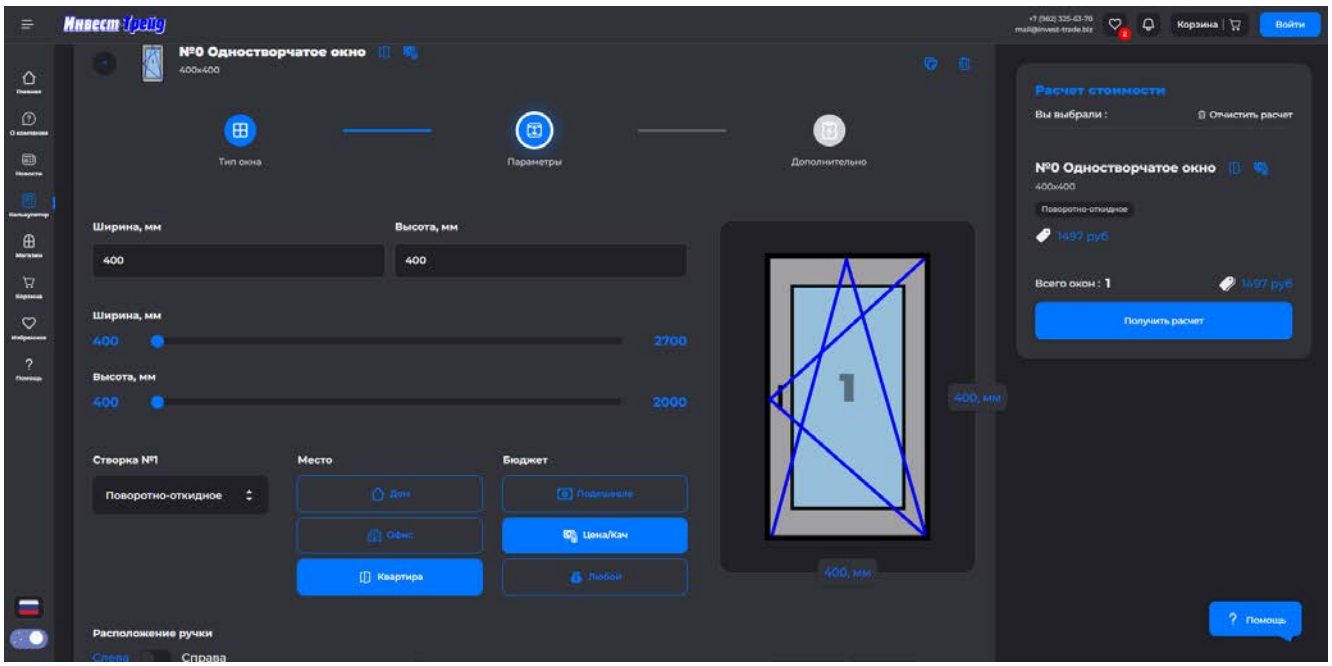

Рисунок 11 – Этап настройки системы

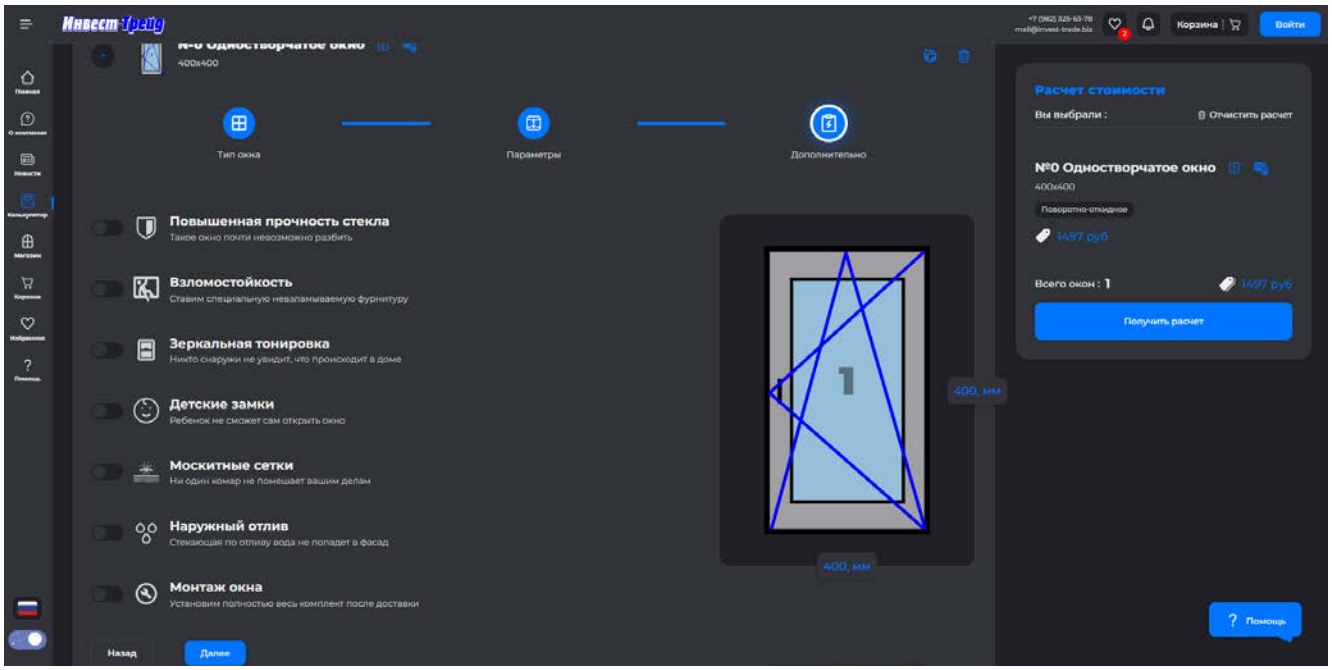

Рисунок 12 – Этап выбора дополнительной комплектации

Для реализации динамической смены схемы пластиковых окон были созданы отдельные векторные картинки для каждого варианта типа, количества створок и их вариантов, расположения ручек или их отсутствие из-за выбора глухой створки.

Необходимо реализовать функцию рендеринга, которая будет принимать в себя тип окна, вид створки и расположение ручки.

Листинг 1 – функция рендеринга изображения схемы

```
const renderIcon = useCallback((sash: Sash, handle: HandleType)
=> {
        switch (sash) {
        case Sash.DEAF:
            return OneStv;

        case Sash.FOLDING:
            switch (handle) {
            case HandleType.LEFT:
                return OneStvLhTop;

            case HandleType.RIGHT:
                return OneStvRhTop;
            }

            return OneStvLhTop;

        case Sash.ROTARY:
            switch (handle) {
            case HandleType.LEFT:
                return OneStvLh;

            case HandleType.RIGHT:
                return OneStvRh;
            }

            return OneStvLh;

        case Sash.SWINGOUT:
            switch (handle) {
            case HandleType.LEFT:
                return OneStvLhTopLeft;

            case HandleType.RIGHT:
                return OneStvLhTopRight;
            }

            return OneStvLhTopLeft;
        }
    }, []);
```

Таким же образом необходимо реализовать функцию смена компонентов этапов расчета пластиковых окон. Создадим состояние с помощью хука useState, в котором будем хранить текущий этап в виде индекса. Далее создадим стрелочную функцию и параметрами будем принимать текущий индекс и возвращать Jsx компонент с помощью оператора switch case. Обернем всю нашу функцию в хук useCallback, который принимает первым параметром обратную функцию, а вторым массив зависимостей. В этот массив передадим переменную расчета для кэширования функции.

Листинг 2 – функция рендеринга изображения схемы

```
const [activeStep, setActiveStep] = useState(0);

const renderBlock = useCallback(
        (activeStep: number) => {
            switch (activeStep) {
            case 0:
                return (
                    <CalculationTypeComponent onChangeType={() =>
{}} type={calculation.type} />
                );
            case 1:
                return (
                    <CalculationParamsComponent
                        calculation={calculation}
                    />
                );
            case 2:
                return (
                    <CalculationAddsComponent
calculation={calculation} />
                );
            default:
                return null;
            }
        },
        [calculation],
    );
```

Функции расчета

Реализуем слайс (slice) в библиотеке Redux для управления состоянием расчета стоимости пластиковых окон в веб-приложении.

Начальное состояние initialState содержит массив calculations, который хранит информацию об одном окне по умолчанию, такую как ширина, высота, тип окна, место установки, бюджет, ручка, дополнительные опции и общую стоимость. Также в начальном состоянии содержатся флаги isLoading и totalPrice.

Определены различные редюсеры (reducers) для управления состоянием приложения:

- CreateCalculation – добавляет новый расчет для окна в массив calculations, при достижении максимального количества расчетов (4) выводит ошибку;
- CopyCalculation – копирует существующий расчет и добавляет его в массив;
- DeleteCalculation – удаляет выбранный расчет из массива calculations, если это не последний расчет;
- setWidth, setHeight – обновляют ширину и высоту выбранного окна, пересчитывая его стоимость;
- setPlace, setBudget, setType – обновляют соответствующие свойства выбранного окна;
- setAdd, removeAdd – добавляют и удаляют дополнительные опции для выбранного окна, пересчитывая общую стоимость;
- setHandle – обновляет тип ручки для выбранного окна;
- setSash – обновляет тип створки для выбранного окна;
- setIcon – обновляет иконку для выбранного окна;
- clearCalculations – очищает все расчеты, оставляя только базовый расчет.

Листинг 3 – код слайса calculationSlice

```
const initialState: CalculationSchema = {
    calculations: [
        {
            width: 400,
            height: 400,
            type: {
                title: 'Одностворчатое окно',
                price: 1497,
                poster: defaultTypePoster,
                sashes: [
                    Sash.DEAF,
                ],
            },
            id: 0,
            place: PlacesType.ROOM,
            icon: OneStv,
            budget: BudgetType.MIDDLE,
            handle: HandleType.NULL,
            adds: [],
            count: 1,
            price: 1497,
        },
    ],
    isLoading: false,
    totalPrice: 1497,
};

export const calculationSlice = createSlice({
    name: 'calculation',
    initialState,
    reducers: {
        createCalculation: (state) => {
            if (state.calculations.length === 4) {
                state.error = 'Максимальное количество расчетов
уже привышено';
            } else {
                state.calculations.push({
                    width: 400,
                    height: 400,
                    type: {
                        title: 'Одностворчатое окно',
                        price: 1497,
                        poster: defaultTypePoster,
                        sashes: [
                            Sash.DEAF,
                        ],
                    },
                    icon: OneStv,
                    id: state.calculations.length,
                    place: PlacesType.ROOM,
                    budget: BudgetType.MIDDLE,
```

```
                    handle: HandleType.NULL,
                    adds: [],
                    count: 1,
                    price: 1497,
                });

                state.totalPrice += 1497;
            }
        },
        copyCalculation: (state, action:
PayloadAction<Calculation>) => {
            state.calculations.push({
                ...action.payload,
                id: state.calculations.length,
            });

            state.totalPrice += action.payload.price;
        },
        deleteCalculation: (state, action:
PayloadAction<Calculation>) => {
            if (state.calculations.length > 1) {
                state.totalPrice -= action.payload.price;
                state.calculations = state.calculations.filter(
                    (obj) => obj.id !== action.payload.id,
                );
            } else {
                state.error = 'Нельзя удалить базовый расчет';
            }
        },
        setWidth: (state, action: PayloadAction<{id: number;
width: number}>) => {
            const { id, width } = action.payload;
            state.calculations[id].width = width;

            state.calculations[id].price = 3 *
(state.calculations[id].width - 400)
             + 5 * (state.calculations[id].height - 400)
             + state.calculations[id].type.price;

            state.totalPrice = state.calculations.reduce((acc,
curr) => acc + curr.price, 0);
        },
        setHeight: (state, action: PayloadAction<{id: number;
height: number}>) => {
            const { id, height } = action.payload;
            state.calculations[id].height = height;

            state.calculations[id].price = 3 *
(state.calculations[id].width - 400)
             + 5 * (state.calculations[id].height - 400)
             + state.calculations[id].type.price;
```

```
            state.totalPrice = state.calculations.reduce((acc,
curr) => acc + curr.price, 0);
        },
        setPlace: (state, action: PayloadAction<{id: number;
place: PlacesType}>) => {
            const { id, place } = action.payload;
            state.calculations[id].place = place;
        },
        setBudget: (state, action: PayloadAction<{id: number;
budget: BudgetType}>) => {
            const { id, budget } = action.payload;
            state.calculations[id].budget = budget;

            if (budget === BudgetType.CHEAP) {
                state.calculations[id].price -= (10 / 100);

                state.totalPrice =
state.calculations.reduce((acc, curr) => acc + curr.price, 0);
            } else if (budget === BudgetType.EXPENSIVE) {
                state.calculations[id].price += (10 / 100);

                state.totalPrice =
state.calculations.reduce((acc, curr) => acc + curr.price, 0);
            }
        },
        setType: (state, action: PayloadAction<{id: number; type:
WindowType}>) => {
            const { id, type } = action.payload;
            state.calculations[id].type = type;
        },
        setAdd: (state, action: PayloadAction<{id: number; add:
IAdds}>) => {
            const { id, add } = action.payload;
            state.calculations[id].adds.push(add);
            state.calculations[id].price += add.price;

            state.totalPrice = state.calculations.reduce((acc,
curr) => acc + curr.price, 0);
        },
        removeAdd: (state, action: PayloadAction<{id: number;
add: IAdds}>) => {
            const { id, add } = action.payload;
            state.calculations[id].adds =
state.calculations[id].adds.filter(
                (item) => item.title !== add.title,
            );

            state.calculations[id].price -= add.price;

            state.totalPrice = state.calculations.reduce((acc,
curr) => acc + curr.price, 0);
        },
```

```
        setHandle: (state, action: PayloadAction<{id: number;
handle: HandleType}>) => {
            const { id, handle } = action.payload;
            state.calculations[id].handle = handle;
        },
        setSash: (state, action: PayloadAction<{id: number;
idSash: number; sash: Sash}>) => {
            const { id, idSash, sash } = action.payload;
            state.calculations[id].type.sashes[idSash] = sash;

            if (sash === Sash.DEAF) {
                state.calculations[id].handle = HandleType.NULL;
            } else {
                state.calculations[id].handle = HandleType.LEFT;
            }
        },
        setIcon: (state, action: PayloadAction<{id: number; icon:
React.VFC<React.SVGProps<SVGAElement>>;}>) => {
            const { id, icon } = action.payload;
            state.calculations[id].icon = icon;
        },
        clearCalculations: (state) => {
            state.totalPrice = 1497;
            state.calculations = [
                {
                    width: 400,
                    height: 400,
                    type: {
                        title: 'Одностворчатое окно',
                        price: 1497,
                        poster: defaultTypePoster,
                        sashes: [
                            Sash.DEAF,
                        ],
                    },
                    id: 0,
                    place: PlacesType.ROOM,
                    icon: OneStv,
                    budget: BudgetType.MIDDLE,
                    handle: HandleType.NULL,
                    adds: [],
                    count: 1,
                    price: 1497,
                },
            ];
        },

    },
});
```

Запрос на сервер для получения результата расчета

Определим асинхронный санк-экшн (thunk action) под названием fetchCalculationResult, который предназначен для отправки данных о расчетах окон на сервер и получения в ответ информации о итоговом продукте.

Здесь используется библиотека createAsyncThunk из пакета reduxjs/toolkit, которая упрощает создание асинхронных экшнов и управление их жизненным циклом.

createAsyncThunk<Product, Calculation[], ThunkConfig<string>> - объявляет тип возвращаемого значения (Product), тип аргумента (Calculation[]) и дополнительный контекст (ThunkConfig<string>).

calculation/fetchCalculationResult – уникальный идентификатор экшна, который будет использоваться при диспатче и подписке на него.

async (data, thunkApi) => { ... } - асинхронная функция, которая будет вызываться при диспатче экшна. Она получает на вход данные data и объект thunkApi, содержащий дополнительные свойства, такие как extra, rejectWithValue.

В теле функции:

Из thunkApi извлекаются extra и rejectWithValue.

Используя extra.api.get выполняется HTTP-запрос на сервер по адресу /products/calculation с передачей массива расчетов data.

Если ответ сервера содержит данные, они возвращаются как результат экшна.

В случае ошибки, с помощью rejectWithValue возвращается сообщение об ошибке, которое будет обработано в редюсере.

Таким образом, этот код определяет асинхронный экшн fetchCalculationResult, который может быть вызван из компонента или другого редюсера для отправки данных о расчетах окон на сервер и получения информации о готовом продукте. Это позволяет интегрировать серверную логику в Redux-приложение и обрабатывать асинхронные операции в едином состоянии.

Листинг 4 – асинхронный санк-экшн fetchCalculationResult

```
export const fetchCalculationResult = createAsyncThunk<
    Product,
    Calculation[],
    ThunkConfig<string>
    >(
        'calculation/fetchCalculationResult',
        async (data, thunkApi) => {
            const { extra, rejectWithValue } = thunkApi;

            try {
                const response = await
extra.api.get<Product>('/products/calculation', {
                    data,
                });

                if (!response.data) {
                    throw new Error();
                }

                return response.data;
            } catch (e : any) {
                return rejectWithValue(e.response.data.message);
            }
        },
    );
```

3.2. Реализация серверной части веб-приложения

Введение.

Разработка современных веб-приложений требует использования надежных и гибких технологий как на стороне сервера, так и на стороне клиента. В этом контексте фреймворк NestJS и база данных MongoDB представляют собой мощную комбинацию, позволяющую создавать высокопроизводительные и масштабируемые серверные решения.

Сочетание NestJS и MongoDB открывает широкие возможности для построения различных типов веб-приложений, от простых API до сложных распределенных систем. Благодаря модульной архитектуре NestJS, разработчики могут легко расширять и масштабировать свои приложения, добавляя новые функциональные возможности без нарушения существующей структуры. Кроме

того, использование TypeScript в NestJS гарантирует более высокую надежность и безопасность кода, что особенно важно для критически важных систем.

В последующих разделах мы рассмотрим подробнее, как использовать NestJS и MongoDB для создания высокопроизводительных серверных приложений, обращая внимание на ключевые аспекты, такие как настройка окружения, определение моделей данных, разработка контроллеров и сервисов, а также развертывание и масштабирование готовых решений.

Модуль и контроллер.

Модуль ProductModule определяет структуру и связи для работы с продуктами. Он импортирует модуль TypegooseModule, который обеспечивает интеграцию с MongoDB с помощью библиотеки Typegoose. Модуль также импортирует TelegramModule и UserModule, предполагая, что в приложении есть другие модули, связанные с Telegram и пользователями. Внутри модуля объявлены ProductService в качестве провайдера и ProductController в качестве контроллера. Таким образом, этот модуль инкапсулирует всю логику, связанную с продуктами, делая ее доступной для других частей приложения через экспортируемый ProductService.

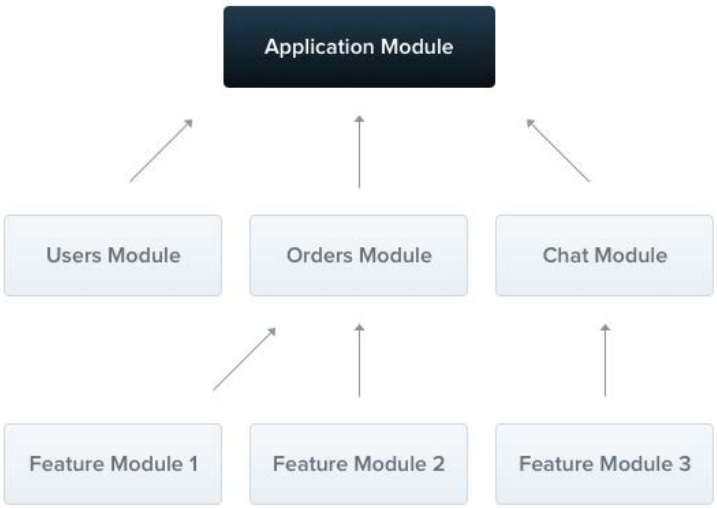

Рисунок 13 – схема подключения дочерних модулей в модуль приложения

Листинг 5 – создание модуля Product для работы с товаром

```
@Module({
  imports: [TypegooseModule.forFeature([{
    typegooseClass: ProductModel,
    schemaOptions: {
      collection: 'Product'
    }
  }]), TelegramModule, UserModule],
  providers: [ProductService],
  controllers: [ProductController],
  exports: [ProductService]
})
export class ProductModule { }
```

Контроллер ProductController определяет набор HTTP-маршрутов для работы с продуктами. Он использует методы, предоставляемые ProductService, для выполнения различных операций, таких как getAll, create, update и delete. Создадим контроллер для получения товара после расчета стоимости калькулятора.

Листинг 6 – контроллер получения товара после расчета

```
@Get('calculations')
  async getCalculationResult(@Body() dto: CreateProductDto) {
    return this.ProductService.getCalculation(dto);
  }
```

Подводя итог по третей главе, можно сказать, что были реализованы основные функции для расчета стоимости пластиковых окон на стороне клиента, создан запрос для оправки данных на сервер. На основе данных, которые пришли в контроллер была написана функция, которая фильтрует товар на окна и характеристики расчета и возвращает их ответом.

ГЛАВА 4. ТЕСТИРОВАНИЕ И ОТЛАДКА ПРОГРАММНОГО КОДА

4.1. Функциональное тестирование

Функциональное тестирование заключается в проверке системы на соответствие заданных в требованиях функциональных задач.

Результаты тестирования приведены в таблице 9.

Таблица 8 – Результаты функционального тестирования

№	**Название теста**	**Шаги**	**Ожидаемый результат**	**Тест пройден?**
1	Заполнить настройки расчета	1. Выбрыть тип окна 2. Ввести ширину и длинну 3.Выюрать створку 3.Выбрать дополнительные характеристики	Цена и картинка должна изменяться динамически в зависимости от выбора	Да
2	Переход по ссылкам меню	1. Просмотреть все пункты меню 2. Нажать на один из пунктов меню	Пользователь должен перейти в соответствующий раздел сайта	Да
3	Открытие сайта на мобильном устройстве	1. Открыть сайт с телефона	Дизайн сайта должен адаптироваться под устройство так, чтобы контент оставался читабельным	Да
4	Возможность добавлять, удалять, копировать окно	1. Нажать на кнопку добавть новое окно 2. Нажать на кнопку удалить новое окно	После нажатия на кнопки должно произойти удаление, добавление или создание нового расчета	Да

4.2. Тестирование верстки

Тестирование интерфейса было проведено в следующих браузерах: Opera, Google Chrome, Microsoft Edge, Safari. Интерфейс пользователя во всех перечисленных браузерах остаётся идентичным, весь контент остаётся на своих

местах. Так же было проведено тестирование верстки на адаптивность в разных браузерах. При увеличении и уменьшении размера экрана контент адаптируется под устройство. Тесты успешно пройдены.

ЗАКЛЮЧЕНИЕ

В рамках данной работы было разработано веб-приложение для расчета стоимости пластиковых окон. При это были решены следующие задачи:

1) выполнен анализ предметной области, выполнен обзор научной литературы;
2) выполнена разработка и проектирование веб-приложения;
3) выполнена разработка серверной и клиентской части веб-приложения;
4) выполнена тестирование и отладка программного кода.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Официальный сайт Veka. [Электронный ресурс]. URL: https://www.veka.ru/ (дата обращения: 14.04.2024 г.).
2. Официальный сайт ОКНА ДАРОМ. [Электронный ресурс]. URL: https://oknadarom74.ru/ (дата обращения: 14.04.2024 г.).
3. Официальный сайт OKNA SERVICE. [Электронный ресурс]. URL: https://www.okna-servise.com/ (дата обращения: 14.04.2024 г.).
4. MongoDB документация. [Электронный ресурс] URL: https://www.mongodb.com/docs/manual/tutorial (дата обращения: 14.04.2024 г)
5. Nest.js документация. [Электронный ресурс]. URL: https://docs.nestjs.com/ (дата обращения: 14.04.2024 г.).
6. React документация. [Электронный ресурс]. URL: https://reactjs.org/docs/getting-started.html (дата обращения: 14.04.2024 г.).
7. TypeScript документация. [Электронный ресурс]. URL: https://www.typescriptlang.org/docs/ (дата обращения: 14.04.2024 г.).
8. Webpack документация. [Электронный ресурс]. URL: https://webpack.js.org/concepts/ (дата обращения: 14.04.2024 г.).
9. Redux документация. [Электронный ресурс]. URL: https://redux.js.org/introduction/getting-started (дата обращения: 14.04.2024 г.).
10. JavaScript документация. [Электронный ресурс]. URL: https://learn.javascript.ru/ (дата обращения: 14.04.2024 г.).
11. HTML документация. [Электронный ресурс]. URL: https://hcdev-ru.pages.dev/html/ (дата обращения: 14.04.2024 г.).
12. CSS документация. [Электронный ресурс]. URL: https://webref.ru/css (дата обращения: 14.04.2024 г.).
13. Node js документация. [Электронный ресурс]. URL: https://nodejs.org/api/all.html (дата обращения: 14.04.2024 г.).
14. Git документация. [Электронный ресурс]. URL: https://git-scm.com/doc (дата обращения: 14.04.2024 г.).

15. RTK query документация. [Электронный ресурс]. URL: https://redux.js.org/introduction/getting-started (дата обращения: 14.04.2024 г.).
16. FSD архитектура документация. [Электронный ресурс]. URL: https://feature-sliced.design/docs/ (дата обращения: 14.04.2024 г.).
17. Jest документация. [Электронный ресурс]. URL: https://jestjs.io/docs/getting-started (дата обращения: 14.04.2024 г.).
18. Storybook документация. [Электронный ресурс]. URL: https://storybook.js.org/docs/get-started (дата обращения: 14.04.2024 г.).
19. Figma документация. [Электронный ресурс]. URL: https://figma.com/ (дата обращения: 14.04.2024 г.).